PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06T 15/70

A1

(11) International Publication Number: WO 98/35320

(43) International Publication Date: 13 August 1998 (13.08.98)

(21) International Application Number: PCT/GB98/00372

(22) International Filing Date: 6 February 1998 (06.02.98)

(30) Priority Data:
9703016.7 7 February 1997 (07.02.97) GB

(71) Applicant *(for all designated States except US)*: CYBERCLASS LIMITED [GB/GB]; Clarendon House, 147 London Road, Kingston–upon–Thames, Surrey KT2 6NH (GB).

(72) Inventors; and

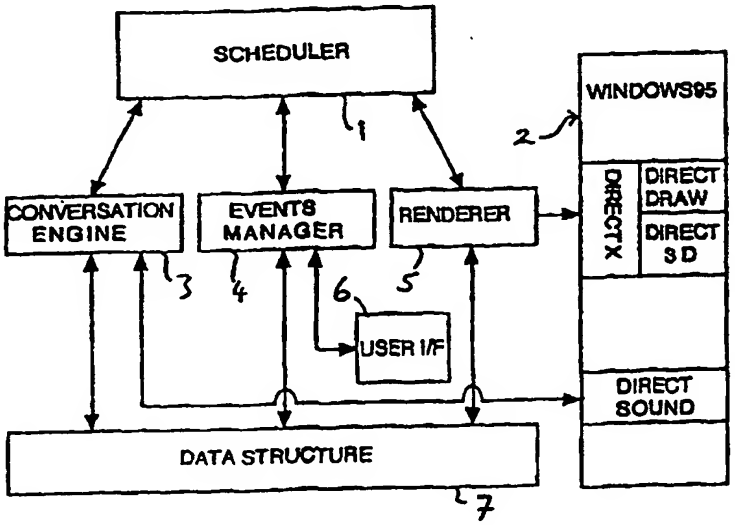
(75) Inventors/Applicants *(for US only)*: RAMSDEN, James, Steven [GB/GB]; Cyberclass Limited, Clarendon House, 147 London Road, Kingston–upon–Thames, Surrey KT2 6NH (GB). MILLS, Christopher, John [GB/GB]; Cyberclass Limited, Clarendon House, 147 London Road, Kingston–upon–Thames, Surrey KT2 6NH (GB). PARKIN, Godfrey, Maybin [GB/GB]; Cyberclass Limited, Clarendon House, 147 London Road, Kingston–upon–Thames, Surrey KT2 6NH (GB). PARKIN, Karen, Olivia [GB/GB]; Cyberclass Limited, Clarendon House, 147 London Road, Kingston–upon–Thames, Surrey KT2 6NH (GB).

(74) Agents: AHMAD, Sheikh, Shakeel et al.; David Keltie Associates, 12 New Fetter Lane, London EC4A 1AP (GB).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: ANIMATION SYSTEM AND METHOD



(57) Abstract

An animation system comprises a scheduler (1) which sequentially triggers a conversation engine (3), an events manager (4) and a renderer (5) during each frame period. Each of the above modules (3 to 5) reads previously written data from and writes data to a data structure (7) which contains a dynamically evolving set of data including hierarchical object data and event data which determines the behaviour of the objects. The objects include avatars which interact with the game player in a 3D environment. The animation system is capable of managing a non–linear interactive storyline such as that used in the playing of an interactive computer game.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

Animation System and Method

The present invention relates to an apparatus for managing and generating animation and to a method of generating an animation sequence.

It is envisaged that such apparatus and method will be useful in the film industry and also in the field of computer games for example.

The invention relates particularly but not exclusively to photo-realistic three-dimensional animation, i.e. to animation which represents three-dimensional movement in a three-dimensional environment scene on a display in a manner approaching photographic realism.

The digital effects industries have produced powerful software packages that allow technically skilled computer artists to create such animation. However such packages are generally too complex for non-technical people with traditional entertainment production talents. Particularly in the computer games industry, this has tended to result in products which lack the creative spark of traditional entertainment media such as motion pictures, television or theatre.

An object of the present invention is to provide an animation system and method which overcomes or alleviates the above problems.

In one aspect the invention provides a system for managing animation comprising a processor arranged to process a dynamically evolving data set including object data and event data, said object data including character data representing avatars, sequencing means arranged to read data from and write data to said dynamically evolving data set in a programmed sequence, said sequence of reading and writing taking place within a frame period and said written data being generated from said read data by stored interactions between said object data and event data, and output means arranged to generate from said dynamically evolving data set a displayable animated output

The system is therefore advantageously capable of providing a real-time animated output

which enables the managing of a non-linear interactive storyline, e.g. for the playing of an interactive computer game. In such an interactive storyline, which is also known as a branching script, the user (or player) can determine how the storyline develops by making selected choices during its progression. The system is also capable of managing a linear storyline, such as is used with a non-interactive computer game, where there are no significant time constraints for the generation of each frame of the storyline.

A method of generating avatars (artificially-generated animated figures in electronic format) suitable for photo-realistic animation is described in our co-pending patent application GB 9610212.4 (a copy of which is filed herewith as an appendix).

Preferably said data set includes stored events, each event comprising a set of one or more preconditions associated with a set of one or more results, and the system includes an events manager controlled by said sequencing means which is arranged to detect said set of preconditions and, on detection of said set of preconditions, to generate said set of results, said output means being responsive to said results to generate said displayable animated output.

For example an event may comprise the set of preconditions:

Bob is at room 1
Fred is clicked
Fred is at room 2

and the set of results:

Move Bob to room 3.

The above result could for example be implemented by a stored animation routine which would move the avatar ABob@ to room 3.

Accordingly the system preferably comprises animation routines arranged to be triggered by said generated sets of results, said output means being arranged to run said triggered animation routines to generate said displayable animated output.

Preferably the system comprises rendering means arranged to render a scene composed of objects associated with said generated sets of results.

Preferably the object data represent a three-dimensional world.

In another aspect, the invention provides a method of generating an animation sequence comprising the steps of generating a dynamically evolving data set including object data and event data, said object data including character data representing avatars, reading data from and writing data to said dynamically evolving data set in a programmed sequence, aid sequence of reading and writing taking place within a frame period and said written data being generated from said read data by stored interactions between said object data and event data, and generating a displayable animated output from said dynamically evolving data set.

Preferably a program containing stored input instructions in a high level language and governing the animation sequence is run and is arranged to modify said dynamically evolving data set.

The invention also provides a computer programmed with a computer game containing an animation sequence generated by a method in accordance with the above aspect of the invention.

Preferred features are defined in the dependent claims.

Preferred embodiments of the invention are described below by way of example only with reference to Figures 1 to 4 of the accompanying drawings, wherein:

Figure 1 is a block diagram showing the architecture of one embodiment of the invention from the standpoint of running a previously created animated game;

Figure 2 is a block diagram showing a hierarchy of object types which can be used in the above embodiment;

Figure 3 is a block diagram of one example of a set of objects which can be used in the

above embodim[redacted] and

Figure 4 is a block diagram showing the architecture of the above embodiment from the standpoint of the creator of an animated computer game.

The above embodiment is based on a Pentium® - equipped personal computer running the Windows95® operating system as indicated by reference numeral 2 (Figure 1).

Referring to Figure 1, the arrangement comprises a conversation engine 3, an events manager 4 and a renderer 5 which are sampled sequentially in that order during each frame period of 1/12 second by a scheduler 1. Conversation engine 3, events manager 4 and renderer 5 are all software modules in this embodiment but could in principle be implemented by dedicated hardware.

In principle the sampling order could be modified by an output of any of the above modules 3 to 5 and accordingly there is provision for two-way data flow (indicated throughout by double-headed arrows) between these modules and scheduler 1. There is also provision for two-way data flow between each of the above modules and a data structure block 7, which contains a dynamically evolving data set including object data and event data, the object data including character data representing avatars.

In use, each of the above modules 3 to 5 reads appropriate portions of the above data set in response to a signal from scheduler 1, processes the above data and then writes the result in the data set. Each module 3 to 5 reads and writes in the above fashion during each frame. Hence events manager 4 reads what conversation engine 3 has written at the beginning of the frame and renderer 5 subsequently reads what events manager 4 has written in the same frame, generates a rendered output (which is sent to the direct X, direct draw and direct 3D modules of Windows 95@ and then output through a port of the computer) before the frame ends. The above sequence is then repeated for subsequent frames.

The conversation engine 3 controls responses of the avatars (Non-Player Characters) to he conversation of the (human) player of the game and has provision for two-way data flow

with the direct sound component of Windows 95®, enabling speech input and sound output via an appropriate sound card, microphone and speaker(s) (not shown). It can generate, and write to the data structure block 7, the precondition components of events which are handled by the events manager 4.

In order to do this, the conversation engine 3 operates on a conversational Anet@ of sentence units, nodes in the net representing pauses in a conversation, where the participants pause, and consider what to say next. The branches from a node represent the possible sentences which could be spoken in response to the sentence which has led to that node (having just been spoken). Furthermore the sentences in the net model not only what is spoken but also events (particularly preconditions to which the events manager 4 is responsive).

It is possible to control any aspect of the game interaction as a result of the way a conversation path through the above net is followed.

All conversations are stored in a comma delimited file, which is loaded into the system at the start of a game. The first line of data details the conversation's parameters, specifically:

Conversation ID (string) Text description

Number of sentences

Number of participants Number of results

First speaker

n participants

n results

Each of the following lines detail a sentence of the conversation:

Sentence ID (integer)

Conversation ID

Cue

Text

Current node

Next node

Nextspeaker

Number of results

n Results

When a conversation is due to start, the list of participants is checked to ensure they are at the same location, aborting if they are not.

The 'node' value is set to zero, (this identifies where the conversation is logically), and the 'Current speaker' is set from the conversation data.
The list of sentences is searched for all sentences with the correct conversation ID and a 'current node' value equal to the 'node' value. If the current speaker is the player of the game, a menu of the sentence cues is offered, or else a sentence is randomly picked. The sentence is then played by a text-to-speech system. Finally, the 'current speaker' and node' values are updated, and the sentence=s results are processed. This loop is then repeated.

When a conversation ends, the conversation's results are played before the last sentence' results, as this allows for varying conversations to be started by different end sentences.

As noted above, the events manager 4 responds to certain preconditions created, for example, by the conversation engine 3 by generating certain results. An Aevent@ is defined as a set of preconditions associated with a set of results and the function of the events manager 4 is essentially to recognise the appropriate sets of preconditions and to generate the appropriate sets of results, which are written to the data structure block 7.

A precondition is some sort of test, to which the result can be TRUE or FALSE. A result is any command which any of the various engines 3 to 5 understand.
A typical event may look like this:

EVENT I

PRECONDITIONS----------> RESULTS

BOB IS AT ROOM1 MOVE BOB TO ROOM3

FRED IS CLICKED

FRED IS AT ROOM2

Each event is associated with an object in the world. When an object receives an 'update' command, it will examine the list of events associated with it, and execute the event with all TRUE preconditions, and with the highest priority. The execution of an event's results is known as firing.

This association between events and object does have one important implication: currently, only objects within a room occupied by a player=s avatar are updated. If an event needed happened in another room, it would only happen when the player arrived.

For example: in a twist to the plot, the head of marketing decides to dynamite the MD's office. The dynamite object has an 'explode' event, with a precondition that it has been in the room for 10 seconds. Until the player is in the MD's office, the dynamite will not be updated. Potentially a lethal situation for the player.
The following information defines an event:

ID

An unique integer between 1 and 999,999,999. This is currently user defined, but could be automated.

Description

A text description of what the event does, that is currently ignored and lost when the window is closed.

Frequency

If the user required an event that will repeat at timed intervals, such as the ticking of a clock, specify the number of 'update loops' which the event will wait for. It is important to check that the delete after fire box is not active, which could be done in software.

Acceptable frequencies range from 0 to 64,000. Note that an event with a frequency of 0 will try to fire on every update.

Priority

An integer from 0 to 99999999, with 0 being the most urgent. If a number of events are ready to fire, the event with the lowest priority figure will fire. Currently, if a number of events have the same priority, then the first one found is fired. Ranges need to be determined for effective use of this feature.

Delete after fire

This tick box is activated if the event is expected to only fire once.

An event can have any number of preconditions in its preconditions section. These determine if the event is able to fire. If there is more than one precondition in an event then they are treated with Boolean AND logic. i.e. they must all be true if the event is to fire. There could be other logical operations available, such as OR, NOT and XOR. If there are no preconditions specified, then the event may always fire.

The following preconditions are currently supported:

[x] IS AT [y]

True if object X's owner is Y.

e.g. Bill IS AT Whitehouse

TRUE

True!

[x] CLICKED

True if object X has been right-clicked in the 3D view.

e.g. Bill CLICKED

RELATION WITH [x] IS +/- [Z]

True if player's relationship with person X is greater or less than Z.

e.g. RELATION WITH Bill + 5

EVENT [x] COMPLETED

True if event X has finished firing.

e.g. EVENT 2468 COMPLETED

TIMEINROOM [x] GT [y] SECS

True if object X has been at its current location for more than Y seconds.

e.g. TIMEINROOM reception GT 10 SECS

CLICKSINROOM GT [x]

True if the user has right-clicked in the current room more then X times.

e.g. CLICKSINROOM GT 2

REMEMBERS [x]

True if the player has been instructed to remember a string X.

e.g.: REMEMBERS SLARTIBARTFART

COLLIDED [what]

This precondition will return TRUE if the given object has collided with something. It is then up to the gameplanner to decide what to do next. A good use would be to combine the precondition COLLIDED with the new RANDOMPLAY result to get various wincing "oohs" and "aahs" from the affected characters (objects) such as avatar heads.

A user interface 6 is provided to enable two-way data flow to and from events manager 4.

The renderer 5 reads the necessary data from block 7 and then generates a rendered video output. The renderer 5 can be as described in our co-pending application referred to above and incorporated herein by reference.

The described embodiment utilises objects which interact with the events handled by the events manager 4. Players, rooms, non-player characters, items of furniture, synthetic displays, and anything else held in the model of the game world are all objects. The system needs to know what is in the world, what each object is in terms of name, location, associated graphics and behaviours, and gameplan events linked to that object.

The events manager 4 can poll each object to determine whether it has an associated event to execute, namely if it has an event to fire. As an event is fired, the events manager 4 routes the results to the appropriate engines 3 to 5 for execution or executes those results itself. The events manager 4 ensures that the correct action is carried out in the correct manner at the correct time by the correct object or engine.

Each object has an associated script file which describes in detail the object's graphics, behavioural attributes, and initialisation and cleanup functionality.

Objects are modelled with consideration to their complexity. More sophisticated object types inherit functionality from simpler types.

Referring now to Figure 2, a hierarchy of object types is shown. The basic object type is an Aartefact@ 8. This divides into Acontainer@ 9 and camera 10 categories. A container 9 could be a room 11 or a person 12, and the latter could be a player 13.

Thus a person is represented as a container 9 plus any person-specific functionality. The container 9 in turn inherits much from artefact 8, notably the basic ability to have a name, a location, and be associated with gameplan events and 3D graphics.

The container 9 adds the functionality of knowing how to store other objects from the gameworld within itself. Room & player have some common functionality and attributes and some distinctive functionality and attributes. For example only rooms have doors and this is an integral part of being a room. Despite this, both people and rooms can contain things (think of a person containing something as a person 'carrying something). So at this

point it is logical to have to separate objects for Room 11 and Person 12 both of which inherit the functionality of containers but redefine their own distinct functionality.

Cameras 10 can be set anywhere in the 3D environment to capture a view of a scene. A stand-alone camera 10 may provide a view from vantage points; a moving camera be directed using movie commands; a POV camera may be fixed to a moving object such as a player for first-person or third-person tracking. Multiple cameras may be used; gameplan event results and movie commands allow complex cinematography.

Referring now to Figure 3, which shows the structure of a group of objects, it will be seen that actual objects as well as classes of objects are stored as hierarchical trees. The root object 14 is an abstract object which is used to link a camera 15 and its associated light 16 with a table 17 supporting a vase 18 which contains flowers 19. The links between these objects mean that they can behave as a group; for example if the table 17 moves the vase 18 moves with it, as do the flowers 19. In fact, in this case the table 17 will always be stationary with respect to the depicted camera 15 (which in turn will be lit in an unchanging manner by light 16) but in principle the table can be viewed by another camera (not shown). A result from an event can be applied to the root object 14 and will then automatically affect all the objects 14 to 19 depending from it. In principle a result of an event could act on the vase 17, for example, to knock it off the table 17, in which case the flowers 19 would be knocked off too.

Referring to Figure 4, which shows the preferred system from the point of view of the creator of the game (the gameplanner) the user populates the data structure 7 with objects and events via an interface 6 which preferably includes a graphical user interface with a pointing device as well as a keyboard. The objects are stored hierarchically as described above with reference to Figure 3 and the events held in the system represent the gameplan.

Each object has one script file associated with it in addition to the object attributes entered into the object's property pages. Script files are held as text data separate from the program. They must have the same name as the object they refer to, they must reside in the SCRIPTS directory and must only be edited with a plain-text editor.

A script file is divided into several sections, with certain sections reserved to do certain things and custom sections designed by the gameplanners. The sections are as follows:

Standard Script File Sections

Script file sections may contain behaviour attributes, event results, movie commands or internal script control commands. Script file sections may be generic for all objects such as initialisation - describing initial settings, common for a particular type of object, e.g. walking for players, or specific to a particular individual object.

The following script file sections are standard:

[INIT] to [ENDINIT]

Load graphics and models.

[POSTINIT] to [ENDPOSTINIT]

Assign graphics to models.

[DELETE] to [ENDDELETE]

Delete graphics.

[BLINK] to [ENDBLINK]

Randomly called random movement. Facial movement is available as standard for players. Usage is optional.

[STARTWALK] to [ENDSTARTWALK]

Start a walking animation.

[WALK] to [ENDWALK]

Loop a walking animation.

[STOPWALK] to [ENDSTOPWALK]

Stop a walking animation.

Custom Script File Sections

Custom sections may be designed for the object which are then referenced from within the script or from gameplan event results. This allows for the creation of object-specific behaviours and attributes under the control of the gameplan or object script.

Script file sections may contain movie commands or event results, which allow for 3D manipulation of the environment or more complex gameplan related actions to be triggered from within a script.

Scripts can be called from the gameplan by two gameplan event results: Playscript and Runscript. The former will execute all the lines of code between a specified start-marker and end-marker in one sweep. Runscript will execute the first line of a script, (presumed to be a SELECT), then execute one line per update. This allows flexible access to the gameplanner to the lowest level properties of an object during the game progression.

Behaviour Attributes are set as flags set in the script file which specify attributes of object behaviour. Whilst the same effects could be duplicated with events, the repetition of creating these events would drive the writers to desperation. In cases of common default behaviours or available behaviours such as blinking, actual code has been provided within the system which can be triggered by the appropriate behaviour attribute flag in the script file, in this case TYPEBLINKER.

TYPEBLINKER Player avatar has the ability to blink, and do so according to the initialised or currently set behavioural parameters.

The following behavioural attribute flags are available:

AAFPLAYER

Specifies the script playing object to be the player's object.

ISSTATIC

Specifies that the script playing object will not move, so only needs to be rendered once when in a third person point of view.

TYPE BOUNCER1

An object of this type will randomly glide around the screen, bouncing off other objects and the sides of the screen. Implemented for screen savers.

TYPE BLINKER1 [N]

Specifies that the object will blink randomly. If no value is supplied then a default value of 20 is used. The chance of blinking is 1 in N. Note that 'blinking' can be any random movement as defined by the actual section in the script file, such as raising of eyebrows, to the swinging of a tail.

TYPE NOBLINKER1

Stops an object from blinking. Same as TYPE BLINKER1 0.

FLOOR

Specifies the object is the floor of a room. If the player collides with the floor, the collision is ignored.

REMOVEFLOOR

Removes the 'FLOOR' status of an object.

Script control commands regulate the processing of the scripts. The following commands are available:

RUNSCRIPT start end

Removes the current script from the controlling objects list of active script parts, and adds the new script part, as specified by the start and end identifiers.

ADDSCRIPT start end

Adds the new script part, as specified by the start and end identifiers to the controlling objects list of active script parts.

KILLSCRIPT start

Removes the script specified by the start identifier from the controlling objects list of active script parts.

[n] command #

Repeats the specified command N times. Not compatible with command +. Executes command, then the command on the next line. Not compatible with [N], and should be used with caution when nearing the end of a script section.

In order to simplify the game authoring process, a movie engine 20 is provided which is operated by movie commands from interface 6 and runs corresponding pre-stored animation sequences, which could have been captured previously from a motion capture device 22. Non-programmers can perform actions in 3D environments using English-like commands. The required movie commands can be executed from object script files or from event results as generated by the events manager described above. The motion capture can be as described in our co-pending earlier application referred to above.

Furthermore, a lip synch engine 21 is provided which generates timing files from sampled speech (.WAV) files (which are in turn associated with sentences in the previously described conversation net) in response to certain types of sound waveform associated with specific lip movements such as the sounds A EE O OO and hard sounds such as T. These timing files are used to call appropriate lip animation routines.

An alternative preferred embodiment of the present invention, which is described below, is similar to the above described embodiment except for the different way in which the events manager 4 operates. Accordingly, to prevent unnecessary repetition, only the differences between the two embodiments are described below.

In the previous embodiment, the events manager 4 polls each object to determine whether any event associated with the object has been fired. However, in the alternative embodiment, the events manager 4 incorporates a state machine map (not shown) for each object to establish how each object is to react to changes in its environmental conditions.

The state machine map establishes a set of different states in which the object can be. The object resides in one state at a time and moves to a different state by the action of an event on the object. Movement from one state to another is initiated by a predetermined change in a set of environmental conditions occurring locally to the object. Other non-predetermined changes in the local conditions have no effect on the object and the object remains in its current state. This is best illustrated by way of the following example:

Object: a character called "Bob";
Current State: Bob is sitting down watching television in his lounge;
Predetermined Change in Environmental Conditions: The telephone in the lounge rings;
Event in Response to Sensing Predetermined Change: Bob gets up, walks over to the telephone, picks up the receiver and starts a conversation;
New State: Bob is having a conversation on the telephone;

Having described the present invention by reference to specific embodiments, it is to be appreciated that the described embodiments are exemplary only and are susceptible to modification and variation without departure from the spirit and scope of the invention as determined by the appended claims. It is also to be appreciated that the invention extends to any novel combination or subcombination disclosed herein, whether claimed or not.

Method and Apparatus for generating Moving Characters

The present invention relates to a method and apparatus for generating moving characters. More particularly the invention relates, in preferred embodiments, to a method and apparatus (particularly a programmed personal computer) for creating real-time-rendered virtual actors (avatars).

Techniques exist for creating photo-realistic avatars in eg arcade games and for cinema production but these normally rely either 1) on 3D laser scanning which is very expensive and produces models that are very demanding of processing power and memory and cannot be run on conventional personal computers (PC's), even those based on the Pentium chip, or 2) on Full Motion Video (FMV), which places filmed sequences in computer-playable formats. Such techniques have the following major disadvantages:

a) the appearance of the characters in FMV is flat and non-immersive
b) the colours in FMV are washed out
c) the development cost of both FMV and laser scanning is high
d) the game play is tedious and linear, restricted to pre-filmed or pre-rendered scenes
e) the ROM demands of both are excessive.

Alternatively, developers intercut occasional pre-rendered scenes using non-photo-realistic dummies, but

a) being non-interactive, they inevitably interrupt gameplay
b) they invariably lack realism
c) very long development times are required.

Furthermore this expedient does not avoid the inflexibility and excessive ROM demands of FMV.

Currently no games developers use real-time photo-realistic 3D avatars in games for the PC platform. It is thought that the reason for this is that current 3D scanning technology is inappropriate for this application. In particular, existing whole body scanners produce models of sizes well in excess of 100,000 polygons, which cannot

therefore be rendered in real time on a PC. A reduction by a factor of 100 in the number of polygons is required for such real-time rendering and the polygon reduction tools which can be used for this purpose do not give acceptable results.

An object of the present invention is to overcomeor alleviate some or all of the above disadvantages.

Accordingly in one aspect the invention provides a method of generating a moving character for a display, comprising the steps of i) providing:

a) a three-dimensional representation of the structure of the character, the structure representation changing in real time to represent movement of the character,
b) a three-dimensional representation of the surface of the character which is mapped onto said structure representation, and
c) a two-dimensional representation of frequently changing portions of said surface,

and

ii) combining said representations a), b) and c) to generate a combined representation of the moving character.

In another aspect the invention provides an apparatus for generating a moving character comprising:

i) memory means arranged to provide:

a) a three-dimensional representation of the structure of the character, the structure representation changing in real time to represent movement of the character,
b) a three-dimensional representation of the surface of the character which is mapped onto said structure representation, and
c) a two-dimensional representation of frequently changing portions of said surface,

and

ii) processing means arranged to combine said representations a), b) and c) to generate a combined representation of the moving character.

Preferred embodiments are defined in the dependent claims.

In its preferred embodiments the present invention enables the creation of fully three-dimensional characters that are capable of moving and behaving in a convincingly human fashion, interacting with their environment on a data-driven level, supporting a variety of traditional photographic and cinematic techniques, such as moving from long-shots to close-ups without break or transition, and which provide lip-synchronisation and sufficient facial expression to enable a wide range of conversation and mood, all within the existing constraints of real-time rendering on PC's.

In preferred embodiments in which the rapidly changing area of the character conveys speech and facial expression, emulation of a range of phonemes and a range of eye and forehead movements is provided for. It has been found that by subdividing the face into upper and lower areas and substituting texture maps accordingly, an acceptably wide range of expressions and mouth movements can be conveyed in an economical fashion.

In such embodiments the two-dimensional representations c) are for example of eye, mouth and preferably also forehead regions of the face. These are preferably photographic in origin. Consequently even slight, simple movements in the head convey a feeling of realism, as the viewer reads their own visual experience into the character in a way that cannot be done with any other illustrative technique.

Because the rapidly changing areas of the character are represented two-dimensionally (and therefore do not involve recalculation of position as the character changes orientation, as would be required in the case of a wire frame model in which changes in expression are generated by movements of polygons beneath the surface), less processing power, memory and bandwidth (if the image is transmitted eg over the Internet) is required.

In one embodiment the production path for the avatars is as follows:

Photograph (in 35mm format) subjects head from front and side viewpoints, for geometry creation, placing a physical right-angled calibration target (a pre-marked

set square) horizontally within the photographed image

Photograph subject head-on running through a fixed list of phonemes and facial expressions.

Photograph subject's body from two viewpoints for geometry creation

Using 3d plotting software (eg Wireframe Xpress, previously used for CAD purposes), plot each individual vertex and polygon by hand to describe the surface structure of the heads.

Image re-touch all of the photographic texture-maps so that they synchronise in colour and lighting (eg using Aldus PhotoShop and various other graphics packages).

Sub-divide geometry and apply texture-maps to appropriate areas.

Create back and front segments for all limb sections, and retouch texture-maps

Define inverse kinematics hierarchy and animate using captured motion (eg using a combination of Flock of Birds, Puppeteer, and 3D Studio)

Feed into a suitable 3D engine (eg MUMM-E) which runs under, for instance, Argonaut's BRender real-time 3D engine.

The result is believably human pseudo-video rendered on the fly at frame rates of 12 - 16 frames a second on an ordinary pentium PC.

A preferred embodiment of the invention will now be described by way of example only with reference to Figures 1 to 3 of the accompanying drawings, wherein:

Figure 1 shows the generation of a structure representation mapped onto a surface representation of a human character shown A) in front view and B) in profile;

Figure 2 shows the above character in front view with the frequently changing portions (forehead, eyes and mouth) used for facial expression shown in different

states, and

Figure 3 is a diagrammatic representation of the apparatus used for the capture of motion of the body of the above character.

The detailed production procedure for heads and faces is as follows:

Stage 1: Photography

I: Format: the camera negative format should be 35mm, so that the application can consistently calculate the camera position.

II: Lens: the focal length of the lens should be ideally around 100m; this allows for perspective flattening effects to be made use of.

III: Lighting: Lighting should be diffuse and evenly balanced across both sides of the face. most pleasing results have been obtained with warmer images. Photographing subjects under tungsten lighting using 100-400 ASA daylight balanced film has proven effective.

IV: Triangulation and calibration: The subject is photographed with a calibration target S as shown in Figure 1A and Figure 1B. The calibration target works well when it runs under the subject's chin, with about 200mm (8 inches) vertical separation. In order for the face to be mirrored correctly the front of the target must be parallel to the face. The Z axis of the target should move back over the subjects left shoulder.

The resulting photographs 1 are subsequently used to generate a three-dimensional representation of the surface of the subject onto which is mapped a polygonal model 2 of the underlying structure of the subject which in use is manipulated in three dimensions by the computer running the finished video game. This process will be explained in detail subsequently .

V: Angles: best results are obtained with photographs taken at right angles, with profile (0 degrees) and full-face (90 degrees). The profile shot should cover the entire head.

VI: Mouth movements: the following movements and expressions should be included:

1: Eyes looking left
2: Eyes looking right
3: Sceptical look
4: Laugh
5: Wide eyes
6: Raised eyebrows
7: Mad/cross
8: Sad look
9: Half blink (sleepy)
10: Full blink
11: "Em" - Phonetic M sound -
12: "eff" - Phonetic F sound -
13: "Oh" - Phonetic O sound -
14: "Yew" - Phonetic U sound -
15: "En" - Phonetic N sound -
16: "Arh" - Phonetic R sound -
17: "Ae" - Phonetic A sound -

With these movements care must be taken to ensure no extreme deformations of the jaw-line occur. Keeping the jaw lightly clenched whilst photographing the mouth movements reduces the risk of this.

Typical expressions are illustrated in Figures 2A and 2B. The forehead, eyes and mouth regions 3, 4 and 5 are recorded two-dimensionally and their different states are stored for use by the finished game program. Such use can involve appropriate combinations of states (eg frown, eyes left, lips compressed) according to the game play eg in response to commands from the user of the game.

Stage 2: Face digitisation

I: Work with the profile (0 degree) photo in the left hand window, and the full-face (90 degree) in the right.

II: Digitise the calibration target S first of all. A discrepancy can occur when minor mismatches between calibration data in the target settings window and pixel locations of the points in the images take place. The symptom is usually a misalignment between the calibration grid and one point of the triangle; often the triangle lies out of the calibration plane by around 15 to 25 degrees. The remedy is to selectively lengthen or shorten one of the dimensions in the target settings window, in quarter to half inch increments, until the triangle lies in the plane of the grid.

III: The plotting of the 3d polygonal structure representation 2 on the photographic representation 1 of the surface is illustrated in Figures 1A and 1B. On the face, the first line to plot is that of the profile. This should extend from the curve of the throat, up through the curve of the chin, over the mouth, up under the nose, through to the bridge of the nose, and up the forehead to the hairline. This line 10 (Figures 1A and 1B) should be made up of 10-12 points, and be an open path.

IV: This first profile line can then be taken into Excel to be straightened in the x-axis. The process for working with .3DP files in Excel is described later.

V: Next, the hairline should be followed down to the base of the left ear, and along the jaw. Again, 10-12 points are likely to be needed to keep this line smooth. This line should join the profile at the bottom of the chin.

VI: The first complex component is the eye. Plot this as a flat plane, with a point in the centre - on the iris. This can be moved back and forth in the profile view to adjust for curvature. Plot open paths along the eyebrow, and down the eyelid. Connect these with faces.

VII: The second complex component is the nose, particularly the underside. Make frequent use of the perspective view tool to ensure that the line of the profile is not distorted. This is easy to do, and results in caricature-like geometry is in the final model. Be especially careful that the points under the nose in the two images relate.

VIII: Build the forehead with a horizontal ribbon running across to the temple, just above the eyebrow. Use as few faces as possible, but ensure that the turn of the angle of the forehead is neither to swept or to sharp.

IX: Follow the pit of the eye, from eyebrow to nose. use this line as a basis for the rest of the front of the face.

X: Extend a ribbon across the mouth, probably no more than two or three points. Use this to create a plane from nose to chin; do not over-model the mouth, as the animated texture maps will not fit the geometry if you do.

XII: Join the rest of the face with triangles. When finished, you will have a completed left hand side to the face.

XIII: The next component is the side and back of head. The entire side and back can be extracted from the profile shot if careful. The first step is to start a new project, and then reverse the position of the photographs, so that the full-face is in the left window, and the profile is in the right.

XIV: Begin by plotting from the centre of the hairline around to the base of the jaw, as close as possible to the ending point of the faces jawline. Finish this line with a point buried within the centre of the head.

XV: Plot the outline of the ear. Use one, or if necessary two, points inside the ear to form faces in the same way as has been done with the eye.

XVI: Build transverse ribbons through the hair, from the centre to the left edge of the head.

XVII: For the back of the head, Drop lines vertically down from the transverse crown-line towards the nape of the neck or the base of the hair. In order to establish which point is which when closing faces, use the select point tool (by clicking on a verter).

XVIII: The final component is the neck and shoulders. The point at which you stop building out from the head will be a decision made in accordance with the structure for the rest of the model. It may be appropriate to limit this to the collar line, or extend the neck further into the throat and clavicle area.

XIX: Start a new project, with the profile in the left window now, and the fullface in the right hand window.

XX: The neck is essentially a cylinder, with sufficient modelling to fit into the base of the head and extend into the head. Three lines of 6 points running around the neck should suffice. Once completed, close faces as before.

Stage 3: Mirroring

I: It is now necessary to mirror the face, head/hair, and neck. The face needs more attention than the other two components, in that we need to move the texture map as well as mirror the geometry.

II: Mirroring involves a dos-command line utility called, unsurprisingly, mirror. The syntax for this is very simply: mirror filename.3dp newfilename.3dp. Perform this function on each component.

III: The mirroring process is not entirely stable. It is now necessary to take the new .3dp files into Excel or another spreadsheet (Excel is known to work) and recalculate certain vertex locations. The problem with mirror is that it will reflect geometry around the project centre point, not along the weld line of the model. In addition, mirror does not move the pixel location of the vertex co-ordinates in the texture map. This results in the texture map being mirrored as well as the geometry, rather than it picking up new co-ordinates from the new geometryis pixel location. The effect is somewhat counter-intuitive!

IV: .3dp files are ASCII based text files, with a structure that can be defined as a space-delimited variable. Excel will import this structure, with some effort.

V: Open the file in Excel. Specify the file type as a text file (with the extensions .txt, .csv and .prn). Add *.3dp to the file extensions list, and select the face file.

VI: Excel will launch one of its "wizard" utilities; on the first page of this, select edelimitedi as being the file type. On the second, turn off "tab" and turn on "space". On the third, select "text" as being the data type.

VII: The file will open. Select all of columns A to J and change the column width to 20. Click on the "centre" icon. Often, an error message will occur saying "selection too large, continue without undo?" Answer yes. This file may now be worked on.

VIII: Approximately one third of the way down the file is the points database, with XYZ co-ordinates for the vertices. The first three point values will relate to the calibration target - ignore them. points 4 to n will be the values for original half of the face; look at the left hand column, which represents the x values. Note two things: first that the first half of the column has negative values, second that the first 10 -12 points havethe same value. These are the first set of points defined for the profile. Make note of this value. Move down the column until the points turn from negative to positive. Thes are the values for the mirored half-face x coordinates. These values must be changed.

IX: The formula for moving these is =SUM(value)-(2*.x), where x is the value from your profile line. Change all the values for the second half of the column using this.

X: At the top of the file is the points database. This gives the values for the equivalent location in x-y pixel co-ordinates for the vertices in left and right pictures. There are four columns (apart from the numbering of the points) They read x-left hand y-left hand x-right hand y right hand. The column to be concerned with is the x-right hand. Again, locate the mirror portion by noting the discrepancies in values, and identify the number given to the weld line. The same process as with the points database needs to be performed, but in this case the formula is SUM(y-value+y), where y is the vlue in pixels from the welding line.

XI: Do the above for both the neck and hair/head, but do not mirror the pixel database - symmetrical hairstyles are easier corrected in 3d Studio.

XII: The next step is to convert the .3dp files to 3d Studio files and assign texture vertices. Again a command line tool, called convert, is used. The syntax is convert filename.3dp -r name of right-hand photograph.bmp matx.tga .

XII: It may well be necessary to work on the texturemap files; the hair may need

to be cloned outwards, the same for the neck in the neck texture, and also clean hair off the forehead by cloning.

XIII: Take files into 3ds, join and move vertices where necessary.

Stage 4: Lip Synching

I: Using the basic fullface image, move to Animator Pro. Load all of the subsequent facial movements as a flic file.

II: Cut out the active area of the face, i.e. around the eyes and down to the mouth, taking care to include areas of skin that might also move.

III: Add these cut-outs in the correct positions down onto the basic face of the subject, using the original fullface shot.

IV: Cut out the areas of the photograph (3, 4 or 5 - Figures 2A and 2B) that will move from frame to frame. For example cut out one of the moving eyes and paste it down onto the original area of the image. Also select some of the surrounding skin and using the C (clip) and L (lasso) tools. Pick up parts of the skin using the C (clip) and M (move) tools. Use the "`" apostrophe mark to paste these into the final flic.

V: The flic file should then contain all of the images from the original photoshoot of seventeen expressions, and if necessary any in-between frames that are required to smooth the facial movement. From this flic file, a render in 3d Studio can ensure that none of the fluid facial movements exceed the limits of the static facial geometry. For final export into Brender or Rendermorphics, individual frames can be generated.

VI: Apply the final animated file (saved in .cel format) to the facial geometry in 3ds. Construct a hierarchy that links the face and the head/hair, and place its pivot point at the nape of the neck. This allows for convincing head movement, combined with photographically realistic facial expressions. Adjust imperfections accordingly.

The body of the character can be animated and combined with the head by standard

techniques. One way of capturing motion of the body is illustrated in Figure 3 wherein magnets are mounted at the joints of the subject's body 8 and their position detected by an electromagnetic positon sensor 8 of standard type.

The avatar derived from such a technique can be controlled and displayed by a computer game.

The invention extends to any novel combination or sub-combination disclosed herein. In particular any photographic representation of the rapidly changing areas of the character, whether or not two-dimensional, may be employed.

Claims

1. A method of generating a moving character for a display, comprising the steps of i) providing:

a) a three-dimensional representation of the structure of the character, the structure representation changing in real time to represent movement of the character,

b) a three-dimensional representation of the surface of the character which is mapped onto said structure representation, and

c) a two-dimensional representation of frequently changing portions of said surface,

and

ii) combining said representations a), b) and c) to generate a combined representation of the moving character.

2. A method as claimed in claim 1 wherein said combined representation is generated in real time.

3. A method as claimed in claim 1 or claim 2 wherein said combined representation is displayed interactively in response to commands from a user.

4. A method as claimed in any preceding claim wherein two or more such two-dimensional representations c) of different frequently-changing changing portions of said surface are generated and are combined with said representations a) and b).

5. A method as claimed in any preceding claim wherein the two-dimensional representation(s) c) is/are of portion(s) of facial expressions.

6. A method as claimed in any preceding claim wherein said three-dimensional surface representation b) is derived from a photograph.

7. A method as claimed in any preceding claim wherein said three-dimensional structure representation a) comprises a multiplicity of polygons.

8. A method as claimed in any preceding claim wherein said combining step ii) is carried out by a program running in a PC and the character is displayed in real time

on the display of the PC.

9. An apparatus for generating a moving character comprising:

i) memory means arranged to provide:

a) a three-dimensional representation of the structure of the character, the structure representation changing in real time to represent movement of the character,
b) a three-dimensional representation of the surface of the character which is mapped onto said structure representation, and
c) a two-dimensional representation of frequently changing portions of said surface,

and

ii) processing means arranged to combine said representations a), b) and c) to generate a combined representation of the moving character.

10. An apparatus as claimed in claim 9 wherein said processing means is arranged to generate said combined representation in real time.

11. An apparatus as claimed in claim 9 or claim 10 wherein said processing means is providedwith user-controlled input means and is arranged to display said combined representation interactively in response to commands from a user.

12. An apparatus as claimed in any of claims 9 to 11 which is arranged to combine two or more such two-dimensional representations c) of different frequently-changing changing portions of said surface with said representations a) and b).

13. An apparatus as claimed in any of claims 9 to 12 wherein the two-dimensional representation(s) c) is/are of portion(s) of facial expressions.

14. An apparatus as claimed in any of claims 9 to 13 wherein said three-dimensional surface representation b) is derived from a photograph.

15. An apparatus as claimed in any preceding claim wherein said three-dimensional structure representation a) comprises coordinates defining a multiplicity of polygons.

16. An apparatus as claimed in any of claims 9 to 15 which is a PC arranged to display the character in real time on its display.

17. An avatar comprising:

a) an animated three-dimensional structure representation ,

b) a three-dimensional surface representation which is mapped onto said structure representation, and

c) a two-dimensional representation of frequently changing portions of the surface of the avatar.

18. A method of generating a moving character substantially as described hereinbabove with reference to Figures 1 to 3 of the accompanying drawings.

Abstract

Avatar generation

An avatar is generated by

i) providing:

a) a three-dimensional representation (2) of the structure of the character, the structure representation changing in real time to represent movement of the character,

b) a three-dimensional representation of the surface (1) of the character which is mapped onto said structure representation, and

c) a two-dimensional representation of frequently changing portions of said surface, eg the portions used to generate facial expressions

and

ii) combining said representations a), b) and c) to generate a combined representation of the moving character. Because the frequently changing portions are represented only two-dimensionally, less processing power and ROM is needed to display the avatar.

Figure 1

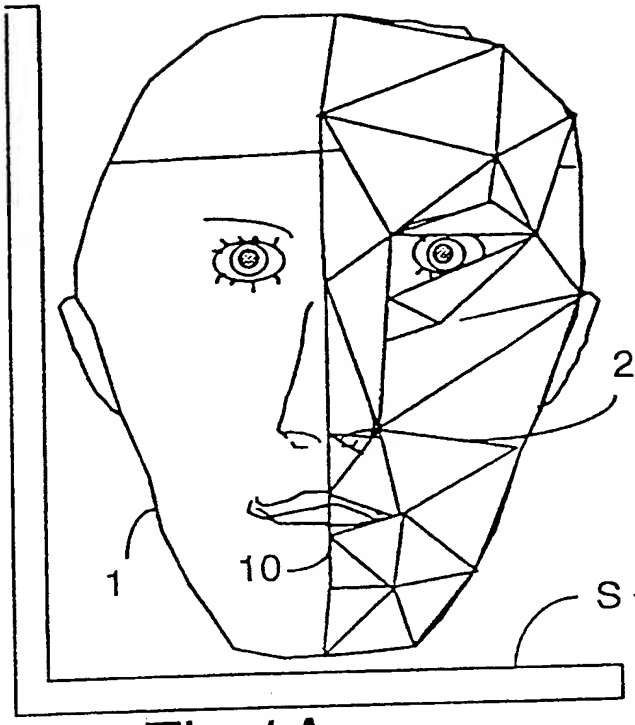


Fig 1A

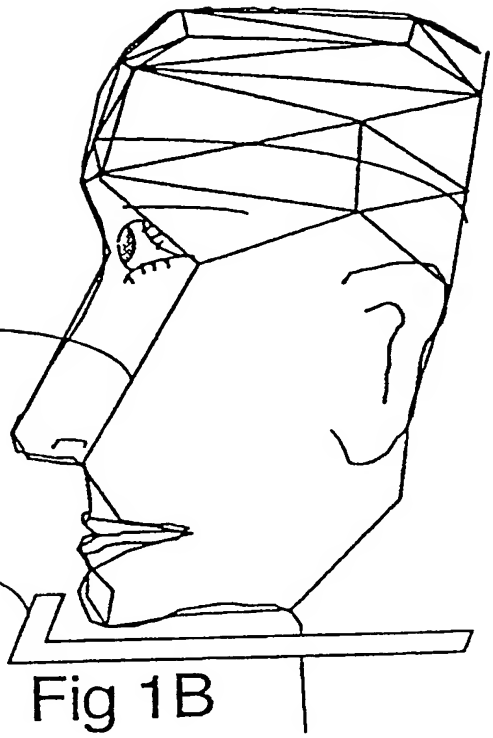


Fig 1B

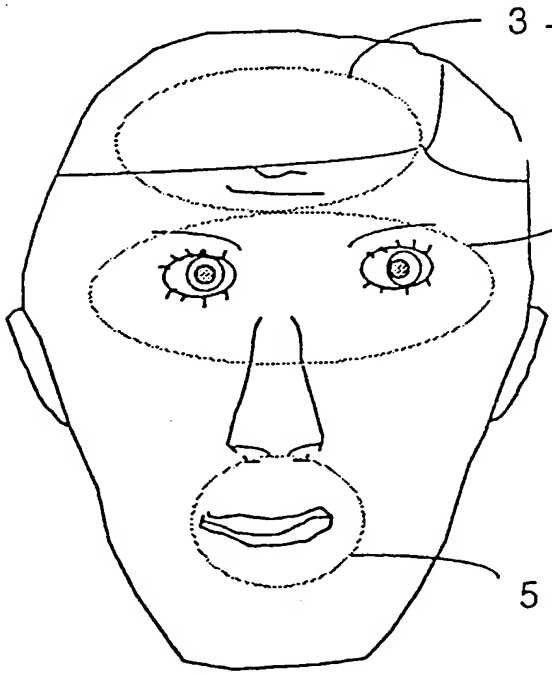


Fig 2A

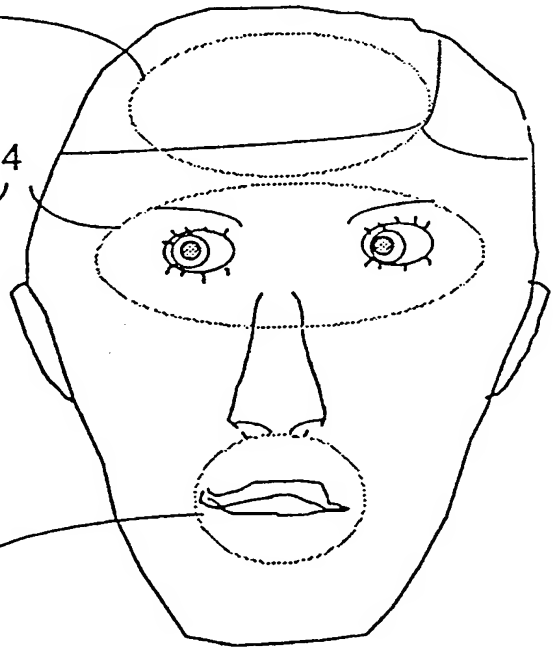


Fig 2B

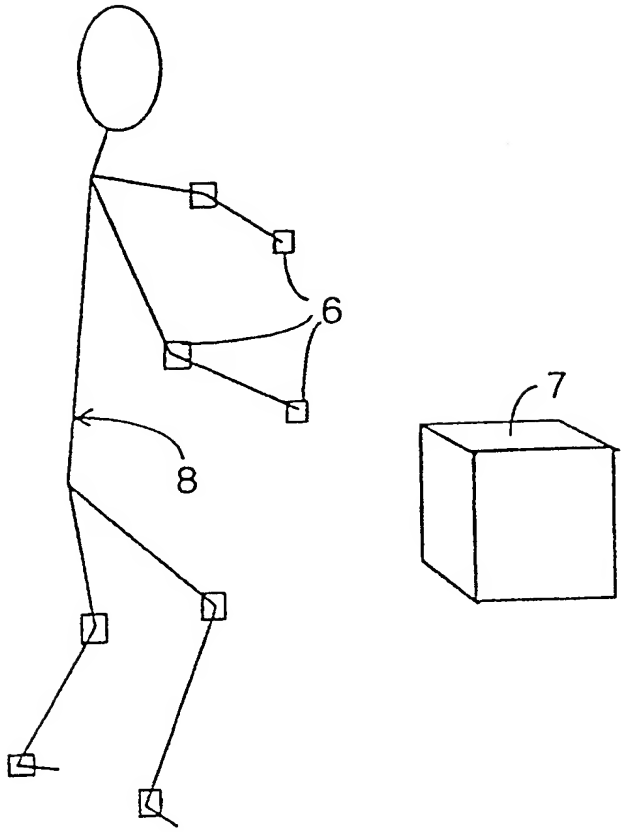


Fig 3

Claims

1. A system for managing animation, the system comprising a processor arranged to process a dynamically evolving data set including object data and event data, said object data including character data representing avatars, sequencing means arranged to read data from and write data to said dynamically evolving data set in a programmed sequence, said sequence of reading and writing taking place within a frame period and said written data being generated from said read data by stored interactions between said object data and event data, and output means arranged to generate from said dynamically evolving data set a displayable animated output.

2. A system according to claim 1, wherein said data set includes stored events, each event comprising a set of one or more preconditions associated with a set of one or more results, and an events manager controlled by said sequencing means is arranged to detect said set of preconditions and, on detection of said set of preconditions, to generate said set of results, said output means being responsive to said results to generate said displayable animated output.

3. A system according to claim 2, wherein the data set includes stored objects, each object having one or more events associated therewith, and said events manager is arranged to poll said objects to establish which events are ready for firing and which objects are associated with the events ready for firing.

4. A system according to claim 2, wherein the data set includes stored objects, each object having one or more events associated therewith, and said events manager utilises a set of state machine maps to determine which objects have events ready for firing and how the object is affected by the firing of those events, each map describing all the possible states a particular object can reside in, the events involved in the movement from one state to another and the predetermined conditions required to trigger said movement from state to state.

5. A system according to any of claims 2 to 4, further comprising animation routines

arranged to be triggered by said generated sets of results, said output means being arranged to run said triggered animation routines to generate said displayable animated output.

6. A system according to any of claims 2 to 5, further comprising rendering means arranged to render a scene composed of objects associated with said generated sets of results.

7. A system according to any preceding claim, wherein the object data represent a three-dimensional world.

8. A system according to any preceding claim, comprising conversation means arranged to generate and respond to dialogue between avatars, said output means being responsive to the output of said conversation means.

9. A system according to claim 8, wherein said conversation means is arranged to process dialogue by traversing a conversational net having linked nodes, the branching from each node representing the possible continuations of the dialogue.

10. A system according to any preceding claim, wherein said object data is in the form of hierarchical trees of object types, each object type having an associated set of parameters and object types at a higher level in the tree having parameters generic to object types to which they are linked at a lower level, object types being distinguished by additional parameters appropriate to their type.

11. A system according to claim 10 wherein said object types include a generic Aartefact@ object type which is linked at a lower level in the tree with a Acontainer@ object type and by a Acamera@ object type.

12. A system according to claim 10 or 11, wherein said object types include a generic Acontainer@ object type which is linked at a lower level in the tree with a Aroom@ object type and by a Aperson@ object type.

13. A system according to any preceding claim, wherein said object data includes data representing groups of associated objects which are physically associated with one another in the animated output, the data representing such groups being a hierarchical tree wherein objects at a higher level in the tree, when subjected to said event data cause objects linked thereto at a lower level in the tree to share at least some aspects of their behaviour.

14. A system according to any preceding claim, comprising a user interface arranged to enable a user to create said object and event data and thereby generate an animation sequence.

15. A system according to claim 14, comprising at least one animation engine responsive to instructions from said user interface to generate animation subroutines and to incorporate them in said data structure.

16. A system according to claim 15, wherein said animation engine includes an input arranged to receive captured motion from motion capture apparatus.

17. A system according to claim 15 or 16, further comprising a lip synchronisation engine coupled to said user interface and arranged to generate lip synchronisation data from input speech, said animation engine being responsive to said lip synchronisation data to synchronise lip movement of the avatars with their speech.

18. A system according to claim 17, wherein stored sampled speech is accessible from said user interface and is arranged to be played in response to keyboard entry of the words or phrases corresponding to the speech.

19. A system according to any preceding claim, wherein said dynamically evolving data set has an input enabling a user to interact with said avatars by modifying said data set.

20. A system according to any preceding claim, wherein said system is arranged to process said dynamically evolving data set to provide a real-time animated output in a format which is suitable for carrying out a non-linear interactive storyline.

21. A system according to claim 20, wherein said system is arranged also to process said dynamically evolving data set to provide an animated output in a format which can be used for carrying out a linear non-interactive storyline.

22. A method of generating an animation sequence, the method comprising the steps of generating a dynamically evolving data set including object data and event data, said object data including character data representing avatars, reading data from and writing data to said dynamically evolving data set in a programmed sequence, said sequence of reading and writing taking place within a frame period and said written data being generated from said read data by stored interactions between said object data and event data, and generating a displayable animated output from said dynamically evolving data set.

23. A method as claimed in claim 22, wherein said data set is initially generated at a user interface as defined in any of claims 14 to 16.

24. A method as claimed in claim 22 or 23, wherein the animation sequence is generated by means of a system as claimed in any of claims 2 to 13.

25. A method as claimed in any of claims 22 to 24, wherein a program containing stored input instructions in a high level language and governing the animation sequence is run and is arranged to modify said dynamically evolving data set.

26. A computer programmed with a computer game containing an animation sequence generated by a method as claimed in any of claims 22 to 25.

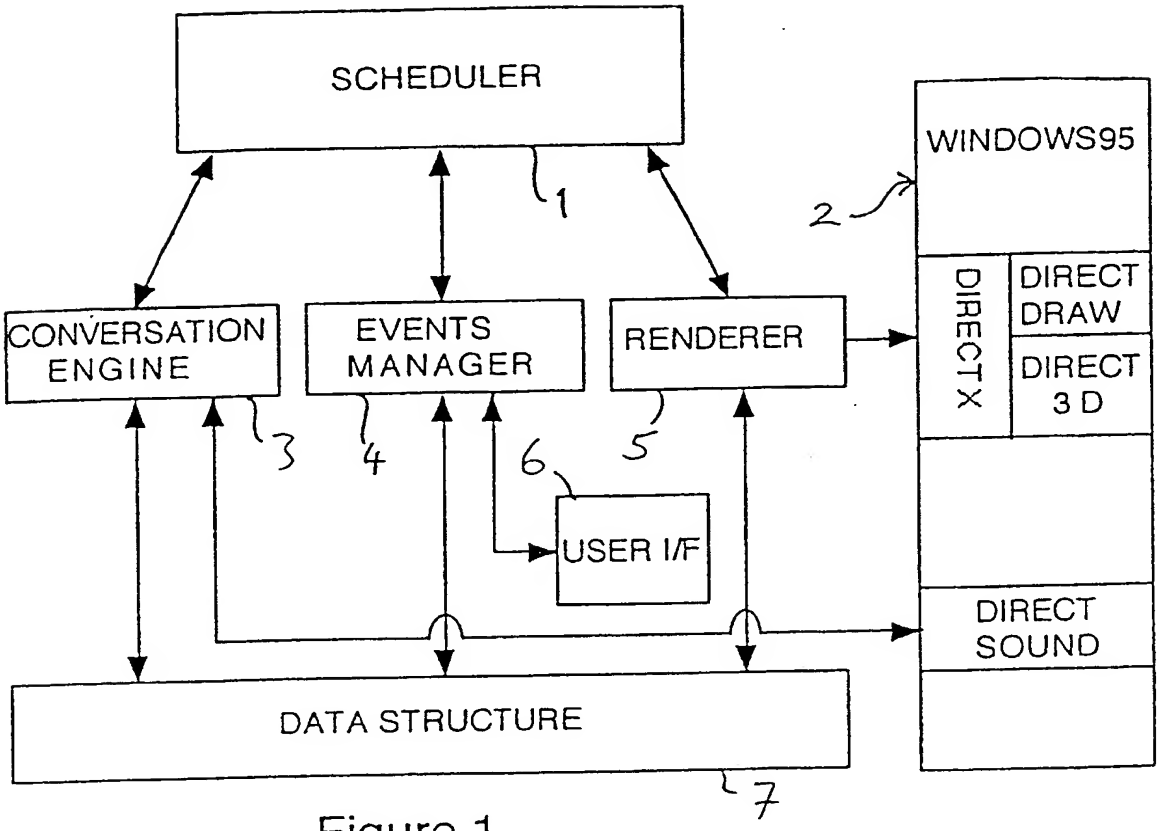


Figure 1

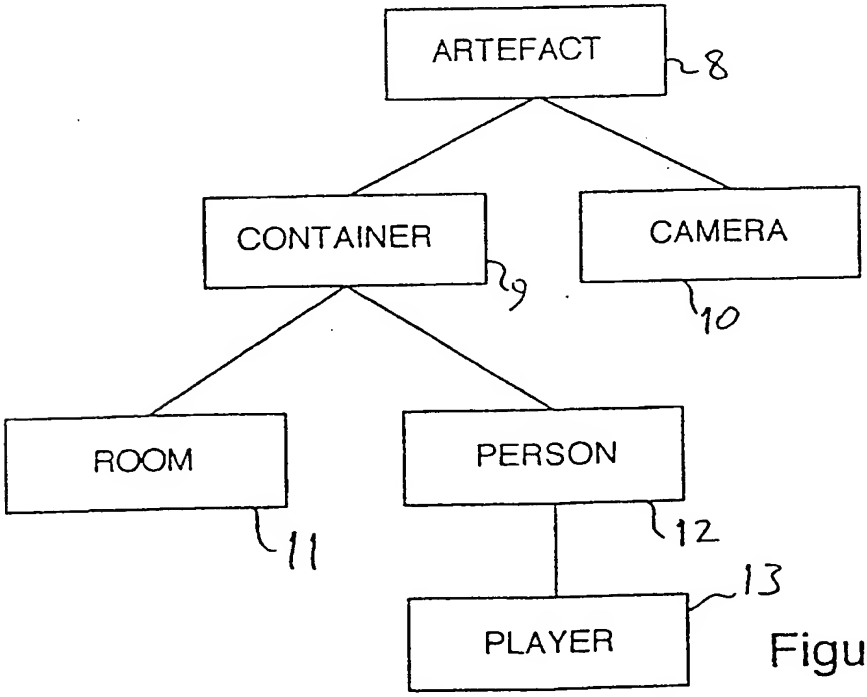


Figure 2

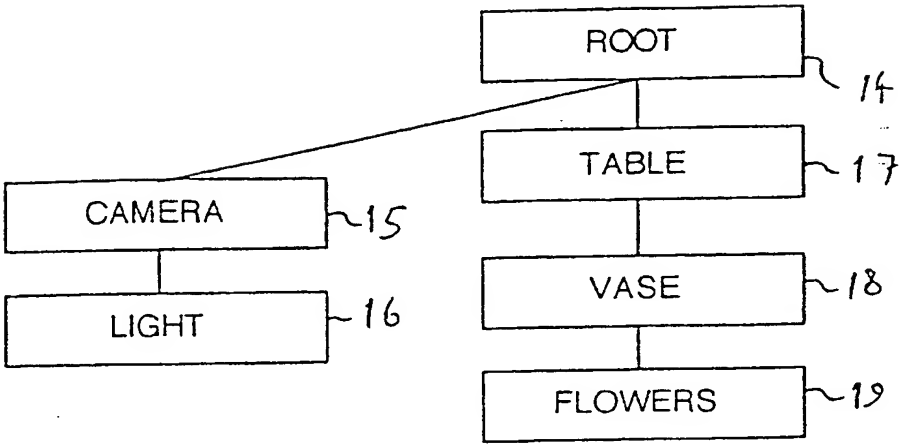


Figure 3

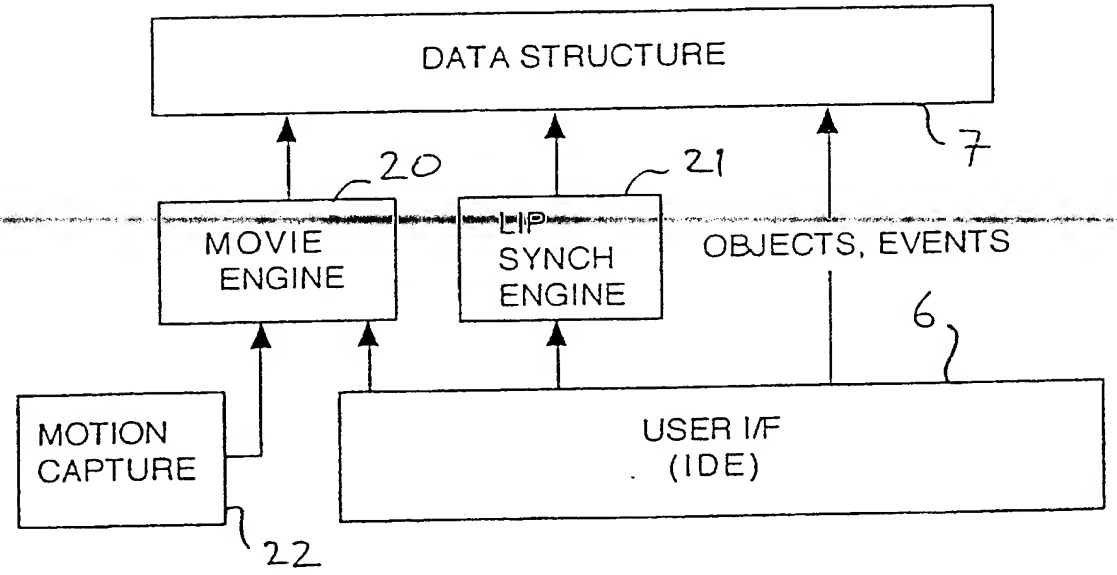


Figure 4

INTERNATIONAL SEARCH REPORT

In. ational Application No
PCT/GB 00372

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06T15/70

According to International Patent Classification(IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 96 23280 A (UNIV COLLEGE OF LONDON ;PENN ALAN RICHARD (GB); DALTON NICHOLAS MI) 1 August 1996 see page 3, line 11 - line 19 see page 4, line 26 - page 5, line 7 see claims 1-3,12	1-3,5-7, 10,14,22
Y	---	4
Y	EP 0 597 316 A (VIRTUAL PROTOTYPES INC) 18 May 1994 see claims 1,5; figure 6	4

X	US 5 596 695 A (HAMADA HIROYUKI ET AL) 21 January 1997 see column 4, line 8 - line 13; claim 1	1-3

	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
19 May 1998	28/05/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Perez Molina, E

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/GB 98/00372

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category '	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	US 4 821 220 A (DUISBERG ROBERT A) 11 April 1989 see column 2, line 55 - column 4, line 11; claim 1	22
A	GAILDRAT V ET AL: "DECLARATIVE SCENES MODELING WITH DYNAMIC LINKS AND DECISION RULES DISTRIBUTED AMONG THE OBJECTS" IFIP TRANSACTIONS B. APPLICATIONS IN TECHNOLOGY, vol. 9, 1 January 1993, pages 165-178, XP000569107 see page 169, paragraph 2.2.; figure 1	1-26
A	STRAUSS P S ET AL: "AN OBJECT-ORIENTED 3D GRAPHICS TOOLKIT" COMPUTER GRAPHICS, vol. 26, no. 2, 1 July 1992, pages 341-349, XP000569106 see page 345, left-hand column, line 26 - page 346, left-hand column, line 26	1-26

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/G[redacted]/00372

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9623280 A	01-08-96	NONE	
EP 0597316 A	18-05-94	US 5485600 A	16-01-96
		CA 2109182 A	10-05-94
US 5596695 A	21-01-97	JP 5274419 A	22-10-93
US 4821220 A	11-04-89	CA 1274310 A	18-09-90
		EP 0254438 A	27-01-88
		JP 63036354 A	17-02-88